Developing a Simulator Application to Test the Billing Platform in Telecom Industry

Mohammad Adel Rostamzadeh Bakhtiyari
School of Information Systems, Queensland University of Technology
Brisbane, Queensland, Australia

m.rostamzadehbakhtiyari@student.qut.edu.au

Abstract

Billing Mediation Platform (BMP) in telecommunication industry is used to process real-time streams of Call Detail Records (CDRs) which can be a massive number a day. The generated records by BMP can be deployed for billing purposes, fraud detection, spam filtering, traffic analysis, and churn forecast. Several of these applications are distinguished by real-time processing requiring low-latency analysis of CDRs. Testing of such a platform carries diverse aspects like stress testing of analytics for scalability and what-if scenarios which require generating of CDRs with realistic volumetric and appropriate properties. The approach of this project is to build user friendly and flexible application which assists the development department to test their billing solution occasionally. These generators projects have been around for a while the only difference are the potions they cover and the purpose they will be used for. This paper proposes to use a simulator application to test the BMPs with simulating CDRs. The Simulated CDRs are modifiable based on the user requirements and represent real world data**.**

INTRODUCTION

Usual telecom applications such as billing are limited to passive analysis and examination of data records. Internet and mobile technologies have presented new ways for communication like Voice over IP (VoIP) calls, mobile web. This has caused development of numerous novel applications including fraud detection, mobile advertising, and spam filtering to exploit data from these communication mediums for achieving different business goals, demand generation and retention, revenue management, and data security. These applications process CDR streams from ongoing calls and take responsive actions in real-time by utilizing complex analytics. The streaming data also has very small lifetime and the sheer amount of their generation (in billions per day) make the traditional store-and-process type of paradigm infeasible. Successful development and consumption of these applications prerequisite a system for very proficient and efficient filtering and processing of call record streams. Both the hardware and software infrastructure constituting the system should be tremendously fast, scalable, flexible and efficient. The development of these applications engages substantial amount of testing for debugging and performance benchmarking. The current BMP infrastructure cannot support streams and hence fall short of supporting these streaming applications. There is thus a need to develop a new architecture for CDR Hubs which can meet the demands posed by emerging streaming applications and services[1]Before moving an application to live environment, the ability and reliability of the application to handle required amount of data and ensure the correctness of its analytics must be verified. Process of validation is often completed by running the application in the production environment via live or pre-recorded data for a trial period of time. The main disadvantage of this approach is that it permits no or very limited control on the data generation, and it does not offer any guarantee that all the available paths in the application's flow are tried within the test duration. For instance a number of of the application's flows may be triggered by unlikely combinations of events, or are particularly rare events, or even hypothetical events used to test what-if-scenarios that have never occurred before. Such event may not be present in the test data and would have to be inserted manually. Furthermore, the application developers may not be authorized to access real data due to privacy concerns. The alternate approach is consequently to test the application via simulated data. Simulated data can be generated randomly according to statistical properties aggregated from actual content. This paper proposes new method to test billing solution platform of small and medium enterprises (SME). This paper is organized as follows section ii provides a literature review on the topic. Section 3 describes the project

and its outcomes. Section 4 explains about the structure of the application and how it relates to other components. Section 5 is about development cycle. Section 6 provides an overview of the whole project.

PROPOSED APPROACH

Companies endeavor either to develop applications in-house or just outsource have to integrate their current enterprise applications such as customer relationship management, provisioning system and billing systems. Testing is an imperative approach to discover errors and bugs in systems development. However, maintaining an effective method of testing can be timely and expensive. Adopting new billing system needs to be supported with the capability of the system to handle the receiving, processing and restoring a huge number of data. Billing system is the backbone of a telecom company and it has direct and indirect impact on company business.without having a robust billing system, it is hard for service provider to offer products and services with attractive deals and ultimately it cannot survive in today's competitive market, accordingly development of the billing system needs to be involved with many testing process. For this reason, tester of the system requires a reliable testing tool to perform the testing and this is the point that the simulator application will be needed to generate real time CDR. Our approach is to develop an application which generates CDR file to test the rating engine as a part of the billing solution with injecting substantial quantity of data sets. This application allows user to modify the settings through its configuration file, including the format of the output and the amount of the output in other word, a dynamic windows form application. This application is based on .NET which is very common in the telecom companies IT infrastructure e

METHODOLOGY

The methodology to develop this project is the Rapid application Development (RAD) .This methodology uses minimal planning and allows the application to be developed faster also makes it easy to change the requirement. RAD is suitable for developing applications where objectives are well defined and narrow with small team of development like this project which is only focused on generating CDR to test other motions.

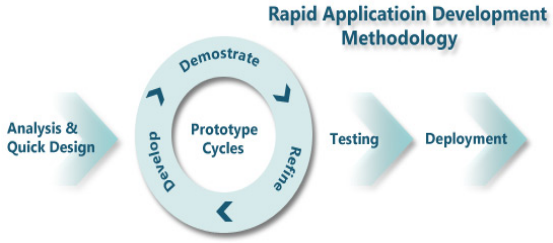

Figure 1 : RAD

DATA FLOW DIAGRAM (DFD)

Figure 2 depicts the data flow diagram of the application and how CDR generator application connects to other components. It only illustrates those entities that related to this project directly

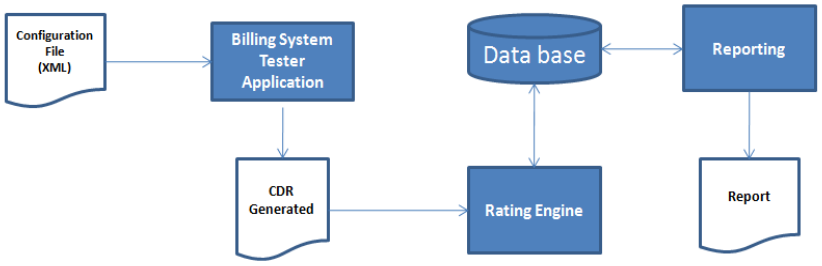

Figure 2 : DFD

SYSTEM DEVELOPMENT

The simulator application has two components. One is the configuration file which is an external XML schema file. This file contains their label name and the value which will be printed in to the CDR file. The second one is CDR generator itself. Figure 3 provides an actual CDR sample which contains a set of different columns such as Customer ID, duration, network, contract and so on.

```
Customer;Contract;Rateplan;Sim_No;Services;MSISDN;Type;
1.10228568;821893;ThurayaPostpay IndigoAustralia;898820
1.10228568;821893;ThurayaPostpay IndigoAustralia;898820
```

Fig 3 : CDR Sample

There are different types of data in CDR file including int or string. The sequence of the columns is also important. Anything that appears in those columns and rows should be under control of the user (tester) of the billing system. Because every service provider in Australia like Telstra or Optus has its own structure of the CDR file so the best solution is to develop a generic flexible application which has ability to fit into any circumstances.

A. Configuration File

This file defines the number of the text boxes, their values and their sequence on the form .by changing the content of configuration file the output will be change accordingly. Every telecommunication company in Australia has its own Figure 4 shows the xml file content.

```
<?xml version="1.0" encoding="utf-8" ?>
<dataRoot>
  <ROWS>
    <labelName>Customer ID:</labelName>
    <TextBoxValue>142536589542241</TextBoxValue>
  </ROWS>
<ROWS>
    <labelName>Date:</labelName>
    <TextBoxValue>15/12/2010-15/12/2011</TextBoxValue>
  </ROWS>
<ROWS>
    <labelName>Date:</labelName>
    <TextBoxValue>142536589542241</TextBoxValue>
  </ROWS>
  <ROWS>
    <labelName>Contract No:</labelName>
```

Fig 4 : Configuration File

Using data root and row elements assist the developer to place the textboxes in order. User only need to type within the tags so application recognizes the instruction from the text file.

Figure 5 shows the interface when the application form reads the configuration file. This is a dynamic interface which changes based on the changes that user provides in the configuration file. Figure 6 illustrates how the number text boxes can be reduces to two.

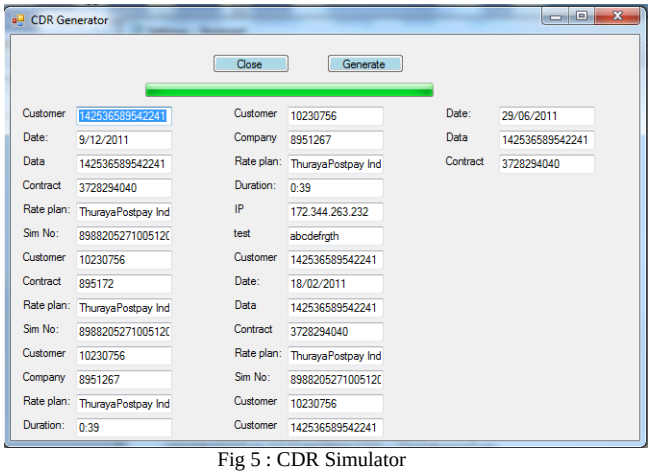

Fig 5 : CDR Simulator

Because this is a dynamic form the button needs to be coded so every time the application runs, it will make the button places on the form in the correct position. Here are the codes lines to define the position and its text on it.

```
Button·btn·=·new·Button();¶
·····btn.Click·+=·new·EventHandler(·btn_Click·);¶
·····btn.Width·=·80;¶
·····btn.Top·=·20;¶
·····btn.Left·=this.Width·/·2·;¶
·····btn.Height·=·20;¶
·····btn.Text·=·"Generate";¶
·····btn.BackColor·=·Color.LightBlue;¶
·····btn.Cursor·=·Cursors.Hand;¶
·····this.Controls.Add(·btn·);¶
¶
```

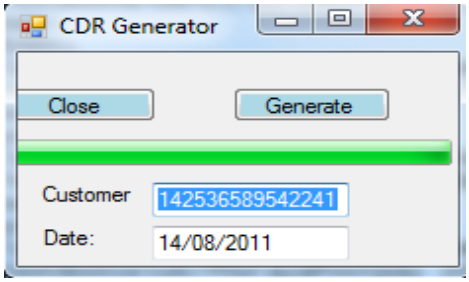

Fig 6 : Flexible Interface

Generating Random Value

A real CDR file contains wide range of data. As a result the simulator application also needs to generator random value such as different duration of call or even different customer ID. These attributes of a CDR file should be modifiable by the user of the system. The following lines of code provide these abilities for the application.

```
DateTime·datetimestart·=·new·DateTime()·,·datetimeEnd·=·new·DateTime();¶
·····Random·r·=·new·Random();¶
·····if·(·value.Contains(·"-"·)·&&·value.Contains(·":"·)·)¶
·····{¶
·····string[]·sss·=·value.Split(·new·string[·2·]·{·"-"·,·":"·}·,·
StringSplitOptions.RemoveEmptyEntries·);¶
·····if·(·sss[·3·]·==·"00"·)·value·=·r.Next(·int.Parse(·sss[·0·]·)·
,·int.Parse(·sss[·2·]·)·)·+·":"·+·r.Next(·int.Parse(·sss[·1·]·)·,·59·);¶
·····else¶
·····value·=·r.Next(·int.Parse(·sss[·0·]·)·,·int.Parse(·sss[·2·
]·)·)·+·":"·+·r.Next(·int.Parse(·sss[·1·]·)·,·int.Parse(·sss[·3·]·)·);¶
·····}¶
·····else¶

·····if·(·value.Contains(·"-"·)·&&·(·DateTime.TryParse(·
value.Split(·new·Char[]·{·'-'·}·)[·0·]·,·out·datetimestart·)·)·&&·(·DateTime.TryParse(·
value.Split(·new·Char[]·{·'-'·}·)[·1·]·,·out·datetimeEnd·)·)·)¶
·····{¶
·····TimeSpan·ts·=·datetimeEnd·-·datetimestart;¶
·····datetimestart·=·datetimestart.AddDays(·double.Parse(·
r.Next(·0·,·ts.Days·).ToString()·)·);¶
·····value·=·datetimestart.ToShortDateString();¶
·····}¶
```

. CONCLUSION

Telecom applications are growing to real-time logic and requiring quick processing of streams of data sets arriving at various rates. The application provides accessible, flexible and manageable testing tool to challenge the billing solution of the company. This application facilitates generating real-time communication data in CSV format so it

can be imported in to a database (rating engine). It is cost efficient in terms of testing the billing solution system. User does not require any programming skills since all the modification happen in the text file.

ACKNOWLEDGMENT

The author would like to thank Indigo Telecom Australia for their support during the development and testing this project.

REFERENCES

[1] 1. Bouillet, E. and P. Dube. *STREAMING WORKLOAD GENERATOR FOR TESTING BILLING MEDIATION PLATFORM IN TELECOM INDUSTRY*. in *Proceedings of the 2010 Winter Simulation Conference*. 2010.

[2] 2. Bouillet, E. and P. Dube. *Streaming workload generator for testing Billing Mediation Platform in telecom industry*. in *Winter Simulation Conference (WSC), Proceedings of the 2010*. 2010.

[3] 3. Anderson, K., et al., *Sword: Scalable and flexible*

[4] *workload generator for distributed stream processing systems.*, in *Winter Simulation Conference*2006.

[5] 4. Beizer, *Software Testing Techniques.* Thomson Computer Press, 1999(2nd edn).

[6] 5. Boyer, Elspas, and Levitt, *a formal system for testing and debugging programs by symbolic execution.* Proceedings International Conference on Reliable Software,, 1975: p. 234–245.

[7]